COCONUT BACK-END API MANUAL

CATALOG

[1.CERTIFICATION 1](#_Toc528763749)

[(1) The first step for certification request 1](#_Toc528763750)

[(2) The second step for certification request 2](#_Toc528763751)

[(3) Revoke 3](#_Toc528763752)

[(4) Get basename 4](#_Toc528763753)

[(5) Get verifier’s group public key 5](#_Toc528763754)

[(6) Recode signatures signed with each basename 7](#_Toc528763755)

[2.SIGNATURE BUSINESS 8](#_Toc528763756)

[(1) Get the relationship between verifiers and KYCs 8](#_Toc528763757)

[(2) Signing transmit 9](#_Toc528763758)

[6.NOTICES 10](#_Toc528763759)

[Appendix 10](#_Toc528763760)

# **1.CERTIFICATION**

## **The first step for certificate request**

**url:** ……/coconut/certificationRequestStep1

**HTTP method:** POST

**Description:** This is the first step for requesting an identity certificate request. Each account can get three certificates in seven days at most.

**Precondition:**

1) User has logined.

2) User has done an identity authentication.

3) User has got less than three certificates in seven days.

**Request json:**

|  |
| --- |
| {  “kyc”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| kyc\* | string | Name of KYCs。 |

**Response json:**

|  |
| --- |
| {  {“errorCode”:”xxx”,  “description”:”xxx”,  “epidPub”:”xxxxxxxxxxxxxxxx”,  “issuerNonce”:”xxxxxxxxxxxx”,  “timeToWait”:”xxx hours”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |
| epidPub | string | The group public key in base64 encoding. |
| issuerNonce | string | Nonce of certificate issuer in base64 encoding. |
| timeToWait | string | Time to wait. |

## **The second step for certificate request**

**url:** ……/coconut/certificationRequestStep2

**HTTP method:** POST

**Description:** This is the second step for requesting an identity certificate request.

**Precondition:**

1) User has logined.

2) User has done the first step successfully during this session.

**Reqeust json:**

|  |
| --- |
| {  “joinRequest”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| joinRequest\* | string | The join request in base64 encoding. |

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”,  “credential”:”xxxxxxxxxxxxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |
| credential | string | Part of user’s identity certificate in base64 encoding. With the data generated in client, it can be used to compose the whole identity certificate. |

## **Revoke**

**url:** ……/coconut/userRevoke

**HTTP method:** POST

**Description:** When user’s identity certificate is stolen, the user can revoke the certificate against the pointed KYC by this API.

**Precondition:**

1) User has logined.

2) User has a valid identity certificate against the pointed KYC.

**Request json:**

|  |
| --- |
| {  “kyc”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| kyc\* | string | Name of KYC. |

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |

## **Get basename**

**url:** ……/coconut/getBasename

**HTTP method:** GET

**Description:** User can get a sort of basenames using this API. A basename is needed when signing a signature.

**Precondition:** null.

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”,  "basenameCount":"xxx",  "basenames":  [  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"},  {"basename":"xxx"}  ]  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |
| basenameCount\* | int | Count of basenames. |
| basenames\* | Json array | A sort of basenames in json array format. |

## **Get verifier’s group public key**

**url:** ……/coconut/getVerifierPublicKey

**HTTP method:** POST

**Description:** Verifier can get it’s group public keys by this API.

**Precondition:** null.

**Request json:**

|  |
| --- |
| {  “verifier”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| verifier\* | string | Name of verifiers. |

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”,  "publicKeys":  {  "kyc1":["pubkey1",  "pubkey2",  "pubkey3",  ......]  }  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |
| publicKeys | Json array | Contains group public keys against the pointed verifier. |

## **Recode signatures signed with each basename**

**url:** ……/coconut/signAllBasenames

**HTTP method:** POST

**Description:** After generating a sort of signatures with each basename, user send the signatures to coconut’s backend by this API. Coconut’s backend will extract some information from the signatures and record it for backtrace when disputes occur.

**Precondition:**

1) User has login.

3) User has a valid certificate without information recorded by coconut’s backend against the pointed KYC.

**Request json:**

|  |
| --- |
| {  "kyc":"xxx",  "signatures":[{xxx"},  {“xxx"},  {" xxx"},  ......]  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| kyc\* | string | Name of KYCs. |
| signatures | Json array | Signatures signed with each basename in json array format. |

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |

# **2.SIGNATURE BUSINESS**

## **Get the relationship between verifiers and KYCs**

**url:** ……/coconut/getBusinessInfo

**HTTP method: GET**

**Description:** User can get the relationship between verifiers and KYCs by this API. When a user want to use a pointed verifier’s service, he or she can know which KYC’s identity authentication should be done.

**Precondition:** null.

**Response json:**

|  |
| --- |
| {  "errorCode ":"xxxxxxx",  "description":"xxxxxxx",  "businesses":[{"verifier1":"kyc1"},  {"verifier1":"kyc2"},  {"verifier2":"kyc2"},  ……]  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode\* | string | [Error code](#errorcode) |
| description\* | string | A simple description of the error code. |
| businesses\* | Json array | Relationship between verifiers and KYCs. |

## **Signing transmit**

**url:** ……/coconut/txRequest

**HTTP method:** POST

**Description:** User can send signature to the pointed verifier by this API. This API just undertakes the task of transmitting signature message. So it doesn’t mean that the signature has been verified when the error code returned is ERROR\_SUCCESS.

**Precondition:**

1) User has logined.

2) User has a valid identity certificate against the verifier.

3) Information in signatures signed with each basename has been recorded successfully by coconut’s backend.

**Request json:**

|  |
| --- |
| {  "verifier":"xxx"  "txData":  {“signature:":”xxxx”,  “msg:":”xxxx”,  “groupId”:”xxxxx”,  “basename”:”xxxxx”  }  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| verifier\* | string | Name of verifiers. |
| txData\* | json | Data of transaction. |
| signature\* | string | Signature in base64 encoding. |
| msg | string | The message signed in base64 encoding. |
| groupId\* | string | Group id of certificate. |
| basename\* | string | The basename used for signing. |

**Response json:**

|  |
| --- |
| {  “errorCode”:”xxx”,  “description”:”xxx”,  “verifierResponse”:”xxxx”  } |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| errorCode | string | [Error code](#errorcode) |
| description | string | A simple description of the error code. |
| verifierResponse | string | Verifier’s response message. |

# **3.NOTICES**

1. **Parameters with \* in this text is indispensable, and parameters without \* is optional.**

# **4.Appendix**

## 1.Error code

|  |  |
| --- | --- |
| **Error code** | **Description** |
| ERROR\_SUCCESS | Operation succeed. |
| ERROR\_SESSION\_INVALID | Invalid session. User should login again. |
| ERROR\_REQUEST\_PARAMETER\_WRONG | Request parameter not correct. |
| ERROR\_REQUEST\_NONEXISTENT\_KYC | The appointed KYC is invalid. |
| ERROR\_REQUEST\_PARAMETER\_INVALID | Request parameter is invalid. |
| ERROR\_NO\_RECORD\_EXIST | There is not record in database. |
| ERROR\_REQUEST\_TOO\_OFTEN | Request too often. |
| ERROR\_PASSWORD\_INVALID | Invalid password. |
| ERROR\_CAPTCHA\_INVALID | Invalid captcha. |
| ERROR\_KYC\_INUSE | Identity authentication has done, no more authentication should be requested. |
| ERROR\_KYC\_CONNECT | Connect to KYC failed. |
| ERROR\_DATABASE\_ERROR | Database error. |
| ERROR\_SERVER\_ERROR | Server inner error. |
| ERROR\_CONNECTION | Connect to verifier failed. |
| ERROR\_SMS\_SEND | SMS sending failed. |
| ERROR\_CAPTCHA\_ERR\_TOO\_MUCH | Too many times for wrong captcha. |

Table 1-error code

## 2.Noun explanation

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | EPID | Enhanced Privacy ID, a cryptographic algorithm proposed by Intel for user authentication. Its main feature is that a public key can correspond to multiple private keys. The algorithm includes both random-based and name-based methods. |
| 2 | KYC | Know your customer, is the process of a business verifying the identity of its clients and assessing potential risks of illegal intentions for the business relationship. In this article, it means KYC service provider. |
| 3 | API | Application programming interface. |